.МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА №  43

ОТЧЁТ

ЗАЩИЩЁН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

### Старший преподаватель                                                               Путилова Н. В.

должность, уч. Степень, звание   подпись, дата                    инициалы, фамилия

ОТЧЁТ О ЛАБОРАТОРНОЙ РАБОТЕ №9

Объектно-реляционные базы данных. Проектирование и создание.

по курсу: ПРОЕКТИРОВАНИЕ БАЗ ДАННЫХ

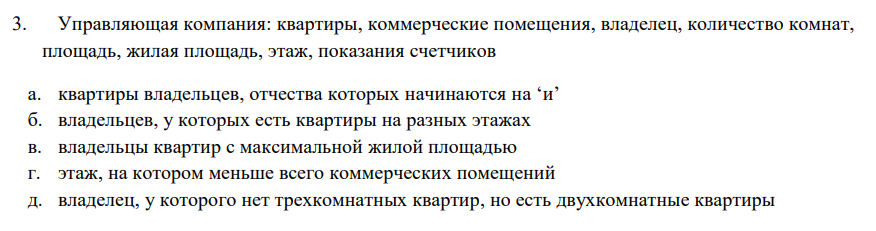
РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. 4136                                                                        Бобрович Н. С.

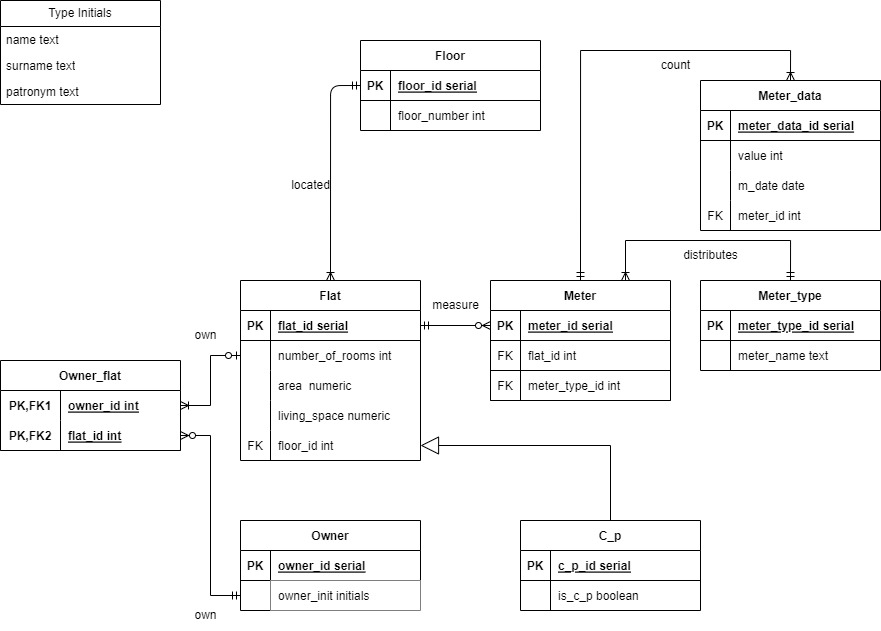
                                                                         подпись, дата                      инициалы, фамилия

Санкт-Петербург 2024

**Вариант 3:**



**Физическая модель БД:**



Скрипт создания БД: (с выводом)

**Создание таблиц и триггеров для их корректного заполнения:**

--CREATE DATABASE db\_1;

CREATE TABLE floor(

floor\_id SERIAL NOT NULL PRIMARY KEY,

floor\_number int

);

CREATE TABLE flat(

flat\_id SERIAL NOT NULL PRIMARY KEY,

number\_of\_rooms int check(number\_of\_rooms > 0),

area numeric check(area > 0),

living\_space numeric check(living\_space > 0),

floor\_id int,

foreign key (floor\_id) references floor(floor\_id) on delete cascade on update cascade

);

CREATE TABLE c\_p (

primary key (flat\_id),

foreign key (floor\_id) references floor(floor\_id) on delete cascade on update cascade,

is\_c\_p boolean

) INHERITS (flat);

CREATE TYPE initials AS (

surname varchar(55),

name varchar(55),

patronym varchar(55)

);

CREATE TABLE owner(

owner\_id SERIAL NOT NULL PRIMARY KEY,

owner\_init initials

);

CREATE TABLE owner\_flat(

owner\_flat\_id SERIAL NOT NULL PRIMARY KEY,

owner\_id int,

flat\_id int,

foreign key (owner\_id) references owner(owner\_id) on delete restrict on update restrict

);

CREATE OR REPLACE FUNCTION check\_flat\_exists\_on\_create()

RETURNS TRIGGER AS $$

BEGIN

IF NOT EXISTS (SELECT 1 FROM flat WHERE flat\_id = NEW.flat\_id) THEN

RAISE EXCEPTION 'flat with id % does not exist', NEW.flat\_id;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER owner\_flat\_check\_flat\_exists\_on\_create

BEFORE INSERT ON owner\_flat

FOR EACH ROW

EXECUTE FUNCTION check\_flat\_exists\_on\_create();

CREATE OR REPLACE FUNCTION check\_flat\_exists\_on\_update()

RETURNS TRIGGER AS $$

BEGIN

IF EXISTS (SELECT 1 FROM owner\_flat WHERE flat\_id = OLD.flat\_id) THEN

RAISE EXCEPTION 'flat with id % сannot be updated because of owner\_flat', OLD.flat\_id;

END IF;

RETURN OLD;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER owner\_flat\_check\_flat\_exists\_on\_update

BEFORE UPDATE ON flat

FOR EACH ROW

EXECUTE FUNCTION check\_flat\_exists\_on\_update();

CREATE OR REPLACE FUNCTION check\_flat\_exists\_on\_delete()

RETURNS TRIGGER AS $$

BEGIN

IF EXISTS (SELECT 1 FROM owner\_flat WHERE flat\_id = OLD.flat\_id) THEN

RAISE EXCEPTION 'flat with id % сannot be deleted because of owner\_flat', OLD.flat\_id;

END IF;

RETURN OLD;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER owner\_flat\_check\_flat\_exists\_on\_delete

BEFORE DELETE ON flat

FOR EACH ROW

EXECUTE FUNCTION check\_flat\_exists\_on\_delete();

CREATE TABLE meter\_type(

meter\_type\_id SERIAL NOT NULL PRIMARY KEY,

meter\_name varchar(55)

);

CREATE TABLE meter(

meter\_id SERIAL NOT NULL PRIMARY KEY,

flat\_id int,

meter\_type\_id int,

foreign key (meter\_type\_id) references meter\_type(meter\_type\_id) on delete restrict on update restrict

);

CREATE OR REPLACE FUNCTION check\_flat\_exists\_on\_create()

RETURNS TRIGGER AS $$

BEGIN

IF NOT EXISTS (SELECT 1 FROM flat WHERE flat\_id = NEW.flat\_id) THEN

RAISE EXCEPTION 'flat with id % does not exist', NEW.flat\_id;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER meter\_check\_flat\_exists\_on\_create

BEFORE INSERT ON meter

FOR EACH ROW

EXECUTE FUNCTION check\_flat\_exists\_on\_create();

CREATE OR REPLACE FUNCTION check\_flat\_exists\_on\_update()

RETURNS TRIGGER AS $$

BEGIN

IF EXISTS (SELECT 1 FROM meter WHERE flat\_id = OLD.flat\_id) THEN

RAISE EXCEPTION 'flat with id % сannot be updated because of meter', OLD.flat\_id;

END IF;

RETURN OLD;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER meter\_check\_flat\_exists\_on\_update

BEFORE UPDATE ON flat

FOR EACH ROW

EXECUTE FUNCTION check\_flat\_exists\_on\_update();

CREATE OR REPLACE FUNCTION check\_flat\_exists\_on\_delete()

RETURNS TRIGGER AS $$

BEGIN

IF EXISTS (SELECT 1 FROM meter WHERE flat\_id = OLD.flat\_id) THEN

RAISE EXCEPTION 'flat with id % сannot be deleted because of meter', OLD.flat\_id;

END IF;

RETURN OLD;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER meter\_check\_flat\_exists\_on\_delete

BEFORE DELETE ON flat

FOR EACH ROW

EXECUTE FUNCTION check\_flat\_exists\_on\_delete();

CREATE TABLE meter\_data(

meter\_data\_id SERIAL NOT NULL PRIMARY KEY,

value int,

m\_date date,

meter\_id int,

foreign key (meter\_id) references meter(meter\_id) on delete cascade on update cascade

);